Aggregation Trees: A Robust Distributed Data Structure for Exascale Computing

Dorian Arnold (U. of New Mexico), Jared Saia (U. of New Mexico) and Barton Miller (U. of Wisconsin).
darnold@cs.unm.edu

Motivation: What happens when we build parallel computers so large that some components are certain to fail during the run of any program? How can we design algorithms and systems to monitor, collect, propagate and analyze data efficiently on such systems? Indeed, these are critical challenges for exascale applications tool and run-time software systems: such infrastructure will generate and analyze unprecedented volumes of data primary to the application or used for secondary services like performance analysis, debugging and resource management.

Our Position: We believe that in order to achieve exascale computing performance, it is compulsory that the research community explore new approaches to fault-tolerance beyond traditional techniques like checkpoint/restart or task replication. In particular, it is necessary to investigate techniques that bear no or low overhead during normal (failure free) operation. The limitations of current techniques have been analyzed and demonstrated in a number of recent research articles, and it is unclear whether the technological advances necessary to render these techniques viable on exascale systems can be available given desirable economic and power budgets.

Specific Problems: Eventually, the HPC community will need determine how to ensure that *any* distributed algorithm can be made reliable, in the sense that it returns the correct output even when multiple processors fail. The determination of "correct output" is based on the semantics and consistency requirements of the application or algorithm. We focus on ensuring reliability in a restricted computational paradigm: *Aggregation Trees (ATs).* ATs enable scalable data gathering and aggregation by enabling any type of distributed computation that can be performed via communication that is restricted to a tree topology. ATs have proven to be useful in many different domains including parallel prefix computations [12], general aggregation services [3, 5, 8, 17], distributed debugging, performance and monitoring tools [14, 18, 19, 20], information management systems [16, 21], stream processing [4] and mobile ad hoc networks (MANETs) [13, 22]. ATs have also been used to improve the scalability and efficiency of higher level applications like image processing [2] and density clustering. At the systems level, ATs are used for booting, system diagnostics, job launch and some file operations. The application focus of ATs is *efficient and scalable data analysis*.

In the AT paradigm, inputs appear as a stream of data at each leaf of the tree, and the output of the desired computation appears at the root. We propose to design efficient and reliable ATs for the following various computational problems including **element count**, **statistical properties** (for example, minimum, maximum and averages), **item frequency**, sets of elements appearing over a certain frequency threshold and **equivalence classification**, where elements are classified according to simple or complex classifiers and algorithms. These computations represent the fundamental building blocks of many complex analysis techniques in HPC application, tool and run-time system domains

We address both *fail-stop faults* and the more challenging *Byzantine faults*, where faulty processors can engage in any kind of deviations from the protocol, including false messages and collusion. For exascale systems, the latter fault model becomes more relevant in light of the concern for silent data corruption and other non-corrected data errors. For efficiency, we demand that the increase in computational resources, when compared with non-reliable aggregation systems, is at most a small multiplicative constant.

For concreteness, we focus solely on reliability in aggregation trees. Our motivation for focusing only on ATs is two-fold: (1) *Utility*: ATs have a proven track record for providing critical services in the HPC domain and (2) *Tractibility*: Tree-based computations are among the simplest class of distributed computations that still have practical applications. We thus believe this is the right class to focus on as a first step to achieving efficient reliability.

Approach and Prior Work: There is no fault-tolerance without some form of temporal or spatial redundancy: either data or the computations that generate data must be replicated. We propose to build on recent algorithmic and mathematical techniques that allow us to break though through some conjectured bottlenecks for reliable computation. We propose a two-pronged approached based on two fundamental concepts: *inherent data redundancies* and *process quoroms*. In both cases, we focus on the algorithmic properties of various computations and use these properties to enable very lightweight, scalable and robust operation.

Leveraging Inherent Redundancies In this context, our central observation is that many AT-based computations naturally maintain redundant state amongst the processes in the system. Intuitively, as information is propagated from the leaves to its root, aggregation state, which generally encapsulates the history of processed information, is replicated at successive levels in the tree. We have demonstrated that the redundant state from processes that survive failures can compensate for lost information, thereby avoiding explicit data replication (like checkpointing or message logs) [1]. Our initial technique's general requirements are that operations be associative and commutative. Our initial work focused on one compensation mechanism, *state composition*. In state composition, orphaned processes propagate their local aggregation state, to their new parents to compensate for any data lost in transit from the orphans to their failed parent or from the failed parent to the failed parent's parent. State composition is appropriate for *idempotent* operations for which re-processing some input elements does not change the computation's output. We have demonstrated that this approach can recover from simultaneous failures in trees with millions of nodes with sub-second recovery latencies and no overhead during failure free operation.

The future proposed work in this area entails exploring other compensation mechanisms as well as approaches to incorporate these new techniques with existing fault-tolerant strategies like checkpoint/restart and replication. For example, we have already theorized about a second compensation mechanism, *state decomposition*, that accommodates non-idempotent operations by precisely computing and compensating only for the lost state. We plan to implement and evaluate the performance and scalability of this approach, which in its current design is not as lightweight as our first mechanism. We also plan to investigate other compensation mechanisms to accommodate larger classes of applications. Another area we plan to explore is the integration with checkpoint/restart and replication techniques. For example, the application processes as the leaves and root of an aggregation tree can processes may be viewed as sequential data sources and sinks amenable to sequential checkpointing, which avoids the non-scalable coordination complexities of distributed checkpointing.

Process Quoroms We propose to build on new algorithmic and mathematical techniques developed in recent results [10] that allow us to break though through some conjectured bottlenecks for reliable computation. These algorithmic and mathematical techniques include: 1) the use of *quorums*, which are small sets of processor, mostly non-faulty, that can work together as single functional units [10, 7]; 2) randomized fingerprinting algorithms that allow us to check the results of a large number of computations efficiently [6]; and 3) a novel distributed technique that ensure that each faulty node can cause only a limited number of corruptions[1] in an amortized sense [11] .

Unfortunately, many of these approaches developed by the distributed algorithms community are not yet practical for HPC, since in order to achieve reliability, they require a blowup in resources that is at least logarithmic in the network size. A novelty of our proposed research is that we will insist on a blowup in resources that is at most a small constant. To achieve this improvement, we plan to circumvent certain efficiency lower bounds by leveraging off 1) the pipelined nature of AT computations; and 2) new *self-healing* algorithms [9, 15, 11] that make local changes to the overlay network topology whenever a faulty computation is detected. Our approach will achieve an amortized efficiency in the following sense: we bound the total computational and communication overhead that any particular faulty processor can cause. We anticipate this research goal can be achieved within a three year time frame.

Key to our approach will be the use of *quorums*: small subsets of processors that work together to form a single, robust functional unit. Assume n is the number of processors in the network and that the fraction of faulty processors is $f < 1/3$. Then a supernode will consist of $C \log n$ processors, selected uniformly at random with replacement from the set of all processors, where C is a tunable constant that controls robustness. The expected number of faulty processors in a supernode will then be $fC \log n$, much less than a majority. If faults in the network occur independently, and there are a polynomial number of such supernodes, then a simple application of Chernoff and union bounds shows that for some value C, the probability that all supernodes have a majority of good processors approaches 1 as n gets large.

- Challenges addressed: This work addresses the exascale OS/R challenges of robustness and resilience in infrastructure for tools, applications and run-time systems.
- Maturity: In both prongs of our approach, we the viability of our approaches have been demonstrated in peer-reviewed publications at selective venues.
- Uniqueness: Exascale systems pose the unique challenge of unprecedented system scales that dramatically exacerbate the need for lightweight fault-tolerance techniques.
- Novelty: Checkpoint/restart and replication based fault-tolerance solutions are the predominant ones in HPC. The proposed solutions offer significantly new directions compared to these traditional approaches.
- Applicability: While we focus on HPC domains, our techniques are generally applicable to any computational environment including "enterprise" computing.
- Effort: This is a multi-year effort that entails more conceptual and design space explorations in the early years and infrastructure development and empirical validations in the out years.

[1]Specifically $O(log^*n)$, where $\log^* n$ is the very slowly growing iterated logarithm function; if n is the number of atoms in the universe, $\log^* n = 6$

References

[1] Dorian C. Arnold and Barton P. Miller. Scalable failure recovery for high-performance data aggregation. In *24th IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2010)*, pages 1–11, Atlanta, Georgia, USA, April 2010.

[2] Dorian C. Arnold, Gary D. Pack, and Barton P. Miller. Tree-based computing for scalable applications. In *11th International Workshop on High-Level Parallel Programming Models and Supportive Environments*, Rhodes, Greece, April 2006.

[3] B.R. Badrinath and Pradeep Sudame. Gathercast: the design and implementation of a programmable aggregation mechanism for the internet. In *9th International Conference on Computer Communications and Networks*, pages 206–213, Las Vegas, NV, October 2000.

[4] Magdalena Balazinska, Hari Balakrishnan, Samuel Madden, and Michael Stonebraker. Fault-tolerance in the borealis distributed stream processing system. In *SIGMOD International Conference on Management of Data*, pages 13–24, Baltimore, MD, June 2005.

[5] Susanne M. Balle, John Bishop, David LaFrance-Linden, and Howard Rifkin. *Applied Parallel Computing*, volume 3732/2006 of *Lecture Notes in Computer Science*, chapter 2, pages 207–216. Springer, February 2006.

[6] A. Broder. Some applications of rabins fingerprinting method. In *Sequences II: Methods in Communications, Security, and Computer Science*, volume 993, page 143152, 1993.

[7] V. Dani, V. King, M. Movahedi, and J. Saia. Brief announcement: Breaking the $o(nm)$ bit barrier: Secure multiparty computation with a static adversary. In *Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing (PODC)*, 2012.

[8] David A. Evensky, Ann C. Gentile, L. Jean Camp, and Robert C. Armstrong. Lilith: Scalable execution of user code for distributed computing. In *6th IEEE International Symposium on High Performance Distributed Computing (HPDC 97)*, pages 306–314, Portland, OR, August 1997.

[9] T.P. Hayes, J. Saia, and A. Trehan. The forgiving graph: a distributed data structure for low stretch under adversarial attack. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 121–130. ACM, 2009.

[10] V. King, S. Lonargan, J. Saia, and A. Trehan. Load balanced scalable byzantine agreement through quorum building, with full information. *Distributed Computing and Networking*, pages 203–214, 2011.

[11] Jeffrey Knockel, George Saad, and Jared Saia. Self-healing of byzantine faults, 2012. Technical Report, University of New Mexico.

[12] Richard E. Ladner and Michael J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27(4):831–838, October 1980.

[13] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *5th Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[14] Aroon Nataraj, Allen D. Malony, Alan Morris, Dorian Arnold, and Barton Miller. A framework for scalable, parallel performance monitoring using tau and mrnet. In *International Workshop on Scalable Tools for High-End Computing (STHEC 2008)*, Island of Kos, Greece, June 2008.

[15] G. Pandurangan and A. Trehan. Xheal: localized self-healing using expanders. In *Proceedings of the twenty-ninth ACM symposium on Principles of distributed computing*. ACM, 2011.

[16] Robbert Van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, 2003.

[17] Phillip C. Roth, Dorian C. Arnold, and Barton P. Miller. MRNet: A software-based multicast/reduction network for scalable tools. In *2003 ACM/IEEE conference on Supercomputing (SC '03)*, pages 21–36, Phoenix, AZ, November 2003. IEEE Computer Society.

[18] Phillip C. Roth and Barton P. Miller. On-line automated performance diagnosis on thousands of processes. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '06)*, March 2006.

[19] Federico D. Sacerdoti, Mason J. Katz, Matthew L. Massie, and David E. Culler. Wide area cluster monitoring with ganglia. In *IEEE International Conference on Cluster Computing*, pages 289–298, Hong Kong, September 2003.

[20] Matthew J. Sottile and Ronald G. Minnich. Supermon: A high-speed cluster monitoring system. In *IEEE International Conference on Cluster Computing (CLUSTER 2002)*, pages 39–46, Chicago, IL, September 2002.

[21] Praveen Yalagandula and Mike Dahlin. A scalable distributed information management system. In *2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '04)*, pages 379–390, Portland, OR, August/September 2004.

[22] Yong Yao and J. E. Gehrke. Query processing in sensor networks. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, January 2003.